Pascal Pons

Follow

SOLVING CONNECT FOUR

Part 8 – Iterative Deepening & Null Window

We will combine in this step two different techniques :

- Iterative deepening increasing iteratively the depth of search while keeping shallow search results in transposition table.
- Null window search using [alpha; beta] window of minimal size (beta = alpha+1) to get faster lower or upper bound of the score.

Iterative deepening

Iterative deepening consists in exploring the search tree at a shallow depth first and then iteratively deeper. It allows to find shallow winning path earlier and can also allow to keep in transposition table the early results of the previous explorations. It helps next iterations to prune the tree exploration more efficiently.

Our specific score function allows to limit the depth of our exploration by limiting the [alpha; beta] search window. Indeed the score of a position is bounded by the number of remaining moves and AlphaBeta algorithm will stop exploring further if it knows the deeper position cannot have a score within the exploration window. For example, if we explore with the window [5; 21] it will explore for a winning path up to 5 moves before the end. And symetrically if we explore with a window [-21; -5], it will explore for a losing path up to 5 moves before the end.

Null window search

This techniques uses a minimal size window [alpha; alpha+1] to test if the score of a position is higher or lower than alpha. An output lower or equal than alpha will tell us that the actual score is lower or equal than alpha. An output higher than alpha will tell us that the actual score is higher than alpha. Having a very narrow windows allow more pruning and faster answer.

We will use these null window explorations in a dichotomic search to find the exact score.

Null window search

Implementation

We implement a new exploration strategy in the solve() function calling the Negamax evaluations. We first compute the maximal window [min, max] of possible score and then iteratively pick a middle point med helping us to narrow the exploration window by running a null window search for this score.

We first favor exploration at shallow depth by running the equivalent of two parallel dichotomic searchs, one for positive scores and one for negative score. This mechanism will iteratively test deeper and deeper positions until it finds either a winning or losing path.

```
int solve(const Position &P, bool weak = false)
{
  int min = -(Position::WIDTH*Position::HEIGHT - P.nbMoves())/2;
  int max = (Position::WIDTH*Position::HEIGHT+1 - P.nbMoves())/2;
  if(weak) {
    min = -1;
    max = 1;
  }
  while(min < max) {                    // iteratively narrow the min-max exploration window
    int med = min + (max - min)/2;
    if(med <= 0 && min/2 < med) med = min/2;
    else if(med >= 0 && max/2 > med) med = max/2;
    int r = negamax(P, med, med + 1);   // use a null depth window to know if the actual score is greater or smaller than med
    if(r <= med) max = r;
    else min = r;
  }
  return min;
}
```

Full source code corresponding to this part.

Benchmark

Exploring at a shallow depth first speeds up considerably the computation of position with a short term outcome (easy and medium test set). It has almost no impact on weak solver because we are using a [-1;1] exploration window that does not benefit much of the new strategy.

Solver	Test Set name	mean time	mean nb pos	K pos/s

Solver	Test Set name	mean time	mean nb pos	K pos/s
Iterative Deepening (strong solver)	End-Easy	7.622 μs	131.6	17,270
Iterative Deepening (strong solver)	Middle-Easy	319.0 μs	9,472	29,690
Iterative Deepening (strong solver)	Middle-Medium	48.30 ms	1,699,000	35,170
Iterative Deepening (strong solver)	Begin-Easy	9.171 ms	236,700	25,810
Iterative Deepening (strong solver)	Begin-Medium	4.817 s	183,600,000	38,120
Iterative Deepening (strong solver)	Begin-Hard	N/A	N/A	N/A
Iterative Deepening (weak solver)	End-Easy	5.255 μs	74.40	14,170
Iterative Deepening (weak solver)	Middle-Easy	1.049 ms	29,910	28,520
Iterative Deepening (weak solver)	Middle-Medium	24.08 ms	801,455	33,280
Iterative Deepening (weak solver)	Begin-Easy	1.113 s	36,350,000	32,650
Iterative Deepening (weak solver)	Begin-Medium	1.751 s	63,590,000	36,320
Iterative Deepening (weak solver)	Begin-Hard	N/A	N/A	N/A

Tutorial plan

SOLVING CONNECT FOUR

Updated: August 26, 2017

Previous Next